Abstract—Current hardware and operating system power management mechanisms use heuristics of a general nature that are largely application software independent, thus treating the application as a black box. The resulting communication disconnect between OS power management policies and application-level quality of service goals can either cause breaches in service level agreements (SLA) when the policies are too aggressive, or reduce energy savings when they are too conservative. In this work, we engage enterprise software in the management of its own performance and power by introducing a middleware that we call SLA Governor. The SLA Governor receives periodic updates on application level performance and the applicable SLA. Based on the available leeway and recent history, the SLA Governor makes, and puts into effect, decisions to raise or lower CPU speeds. Compared to state of the art OS based power management, the SLA Governor provides 8.3% better energy savings for an online transaction processing workload (TPoX) using a latest generation machine employing Intel E5-5600 (“Westmere”) series processors.

I. INTRODUCTION

Enterprise servers consume escalating amounts of energy for powering and cooling the computing hardware ([1], [7]). For this reason, power management mechanisms are becoming commonly available in hardware, firmware and operating system (OS) layers. Commonly such approaches employ CPU utilization or proxies thereof as indicative of demand, and do not factor in application information. There are several pitfalls with this approach. First, application performance delivered at the granularity of a complex transaction depends not just on CPU usage but also on speeds of devices. Secondly, when an OS policy is too conservative far less than ideal amount of energy can be saved, and conversely, a machine can fail to deliver SLA mandated performance when the OS policy is too aggressive. At both extremes, it is desirable to reduce utilization and power while minimizing departure from service quality objectives.

In this paper, we propose a power-performance agent called SLA Governor, which exploits the slack between the SLA and the observed application performance, to drive system-level energy savings by directing hardware (HW) course corrections. Since server applications, such as database management systems (DBMS), web-application servers etc, typically monitor various metrics for self regulating capabilities, our Governor does not require any modification to the application, but rather takes as input the database-monitored metrics. We explore the effectiveness of our system using an Intel platform from the E5-5500(Nehalem) with an enterprise workload called TPoX (an OLTP workload). Using the default Linux OnDemand ([6]) power governor as the baseline in our evaluation, we obtain 8.3% additional savings in system energy for the TPoX workload.

II. RELATED WORK

Service-level agreements (or other metrics of Quality-of-Service) have been widely explored as a mechanism to drive power management in various types of systems, for example real-time systems ([3]), and web-servers ([8]). The closest to our work is SLA-based hardware correction in [4]. However, the approach of [4] results in application level performance oscillations and SLA violations as it is not sufficiently nuanced or reactive in adjusting its policies.

III. SLA GOVERNOR: AN SLA-AWARE ENERGY OPTIMIZATION FRAMEWORK

Figure 1 shows a block diagram of the SLA Governor. The Governor runs as a user-level daemon in the background, which waits for input from the higher-level application by listening on a socket, where the application periodically communicates its performance. The Governor consists of modules which make power optimization decisions and implement them through hardware course corrections. The module Trend Detector processes a number of response time samples over a moving window and smoothens sample-to-sample variations to identify the current performance trend. Its effectiveness is subject to two key parameters: the length of the window, and the averaging method over the window; our experimentation suggested that either arithmetic mean or median taken over a moving window of between four and ten samples provided a good balance between responsiveness and noise reduction. The module depicted as Estimates Desired Performance computes the performance change necessary (delta) to cover the gap between application performance and SLA. The Performance Shaper module uses certain heuristics to adjust the delta; this is in order to avoid wide oscillation in performance or superfluous nudging of hardware. Once the new desired performance delta is determined, the SLA Governor decides the resource allocations which meet the SLA while minimizing the systems power consumption in the Map Performance to HW Resources module. The module shown as Apply New settings enforces the target HW configuration by transitioning processors into appropriate P- and C-states. We use Linuxs system interface to implement these operations.
IV. EVALUATION OF THE SLA-GOVERNOR

Workloads. Transaction Processing over XML (TPoX) is an XML database benchmark simulating a brokerage company for online trading of stocks. We introduced variable think times between transaction requests, such that the system is stressed at random steps of a utilization between 20% and 90%.

Workload Setup and Measurement. Due to space limitations, we only briefly describe our system setup. We use SuSE Enterprise Linux 11.1 with kernel upgraded to version 2.6.37, due to its tickless scheduler, and IBM DB2 9.7 as the database management system that is exercised by the variable think-time TPoX workload described above. We use a 100 GB XML files dataset for TPoX. The TPoX server is equipped with instrumented power supplies and a power management engine called Intel Intelligent Power Node Manager ([5]) which permits the collection of system power readings. In addition, we developed scripts that accumulate statistics about the P- and C-state residencies for processor cores. For SLA, we chose (arbitrarily) a response time of 6 ms, as that was the average response time measured at peak utilization.

A. Results

Figures 2(a) and 2(b) show respectively the response-time and power variations for TPoX under the two governors (baseline or the OnDemand Governor, and SLA Governor). The SLA Governor achieved 8.3% better system-level energy savings while maintaining or improving upon SLA mandated response time. The power savings can be explained by observing that, although the cores spent less time idling, the cores operated for longer durations at lower frequencies. By comparison the OnDemand governor tended to choose highest frequencies or quiescence. This effect has been studied further in [2]. The SLA Governor incurred 291 SLA violations compared to 479 under OnDemand, 39% fewer violations. The reason is simply that SLA Governor is guided strictly by the performance need of the application instead of its CPU utilization.

Figure 2(c) compares the average CPU frequencies over the duration of the TPoX run, between SLA Governor and the baseline. We see that SLA Governor maintains a higher CPU frequency when there is pressure on response time (area in figure labeled with circle 1) and maintains very low CPU frequency when response time is significantly below the SLA (labeled with circle 2). Interestingly, during the initial phases of the run, the workload is mostly I/O bound and has relatively low CPU utilization. Since the SLA Governor is guided by the application demands and not by CPU utilization, it maintains a high frequency and is able to avoid many SLA violations that the baseline was unable to skirt.

Core Consolidation. Unfortunately, the performance impact of core consolidation on application performance proved much more difficult to predict, due to latencies related to exiting sleep states of idle cores, caching effects, etc. In order to mitigate response time spikes, we modified the SLA Governor to provide temporary frequency boosts to the remaining cores, but in doing this, we were precluded from achieving deeper energy savings in our setup, where the transitions between low and high performance demand were a little too frequent to permit appreciable power savings from core consolidation.

V. CONCLUSION

In this paper, we motivated engaging applications actively in management of their energy footprint, by implementing an SLA-aware power management agent. Compared to a well tuned, state-of-the-art Linux power management governor, we showed that SLA-aware power management can achieve energy savings up to 8.3% for TPoX while meeting or exceeding the SLA mandated performance targets.

REFERENCES